单选

1. 多线程应用程序开发中，（）语句使线程 thread1,进入非运行状态，不再得到 CPU 的时间

   A) Thread thread1=new Thread(this);

   B) thread1.sleep(3000);

   C) thread1.resume();

   D) thread1.start();

解析:

sleep()

在指定的毫秒数内让当前正在执行的线程休眠（暂停执行），此操作受到系统计时器和调度程序精度和准确性的影响。

start()

使该线程开始执行；Java 虚拟机调用该线程的 run 方法。

suspend()

已过时. 挂起线程。具有固有的死锁倾向.

resume()

已过时。 该方法只与 suspend() 一起使用，但 suspend() 已经遭到反对，因为它具有死锁倾向。

run()

使用实现接口 Runnable 的对象创建一个线程时，启动该线程将导致在独立执行的线程中调用对象的 run 方法。你的代码应该写到这.

2. 如何在文件中查找显示所有“*”打头的行（）

   A) find * file

   B) wc -l * < file

   C) grep -n * file

   D) grep '^*' file

解析:

md 正则表达式&转义符

grep [-abcEFGhHilLnqrsvVwxy][-A<显示列数>][-B<显示列数>][-C<显示列数>][-d<进行动作>][-e<范本样式>][-f<范本文件>][--help][范本样式][文件或目录...]

-a 或 --text：不要忽略二进制的数据。

-A<显示行数> 或 --after-context=<显示行数>：除了显示符合范本样式的那一列之外，并显示该行之后的内容。

-b 或 --byte-offset：在显示符合样式的那一行之前，标示出该行第一个字符的编号。

-B<显示行数> 或 --before-context=<显示行数>：除了显示符合样式的那一行之外，并显示该行之前的内容。

-c 或 --count：计算符合样式的列数。

-C<显示行数> 或 --context=<显示行数>或-<显示行数>：除了显示符合样式的那一行之外，并显示该行之前后的内容。

-d <动作> 或 --directories=<动作>：当指定要查找的是目录而非文件时，必须使用这项参数，否则 grep 指令将回报信息并停止动作。

-e<范本样式> 或 --regexp=<范本样式>：指定字符串做为查找文件内容的样式。

-E 或 --extended-regexp：将样式为延伸的普通表示法来使用。

-f<规则文件> 或 --file=<规则文件>：指定规则文件，其内容含有一个或多个规则样式，让 grep 查找符合规则条件的文件内容，格式为每行一个规则样式。

-F 或 --fixed-regexp：将样式视为固定字符串的列表。

-G 或 --basic-regexp：将样式视为普通的表示法来使用。

-h 或 --no-filename：在显示符合样式的那一行之前，不标示该行所属的文件名称。

-H 或 --with-filename：在显示符合样式的那一行之前，表示该行所属的文件名称。

-i 或 --ignore-case：忽略字符大小写的差别。

-l 或 --file-with-matches：列出文件内容符合指定的样式的文件名称。

-L 或 --files-without-match：列出文件内容不符合指定的样式的文件名称。

-n 或 --line-number：在显示符合样式的那一行之前，标示出该行的列数编号。

-q 或 --quiet 或--silent：不显示任何信息。

-r 或 --recursive：此参数的效果和指定"-d recurse"参数相同。

-s 或 --no-messages：不显示错误信息。

-v 或 --revert-match：显示不包含匹配文本的所有行。

-V 或 --version：显示版本信息。

-w 或 --word-regexp：只显示全字符合的列。

-x --line-regexp：只显示全列符合的列。

-y：此参数的效果和指定"-i"参数相同。

拓展阅读:Linux 命令大全

http://www.runoob.com/linux/linux-command-manual.html

3. 在 WEB-INF 目录下，必须存放的文件为（）

A) class 文件

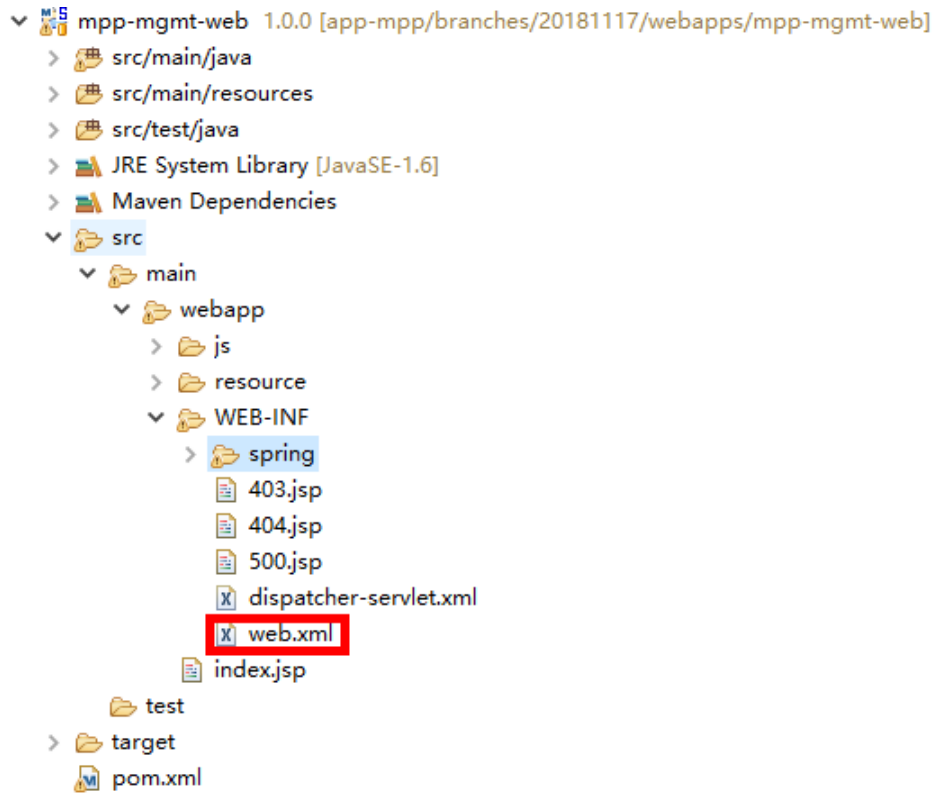
B) web.xml

C) jar 文件

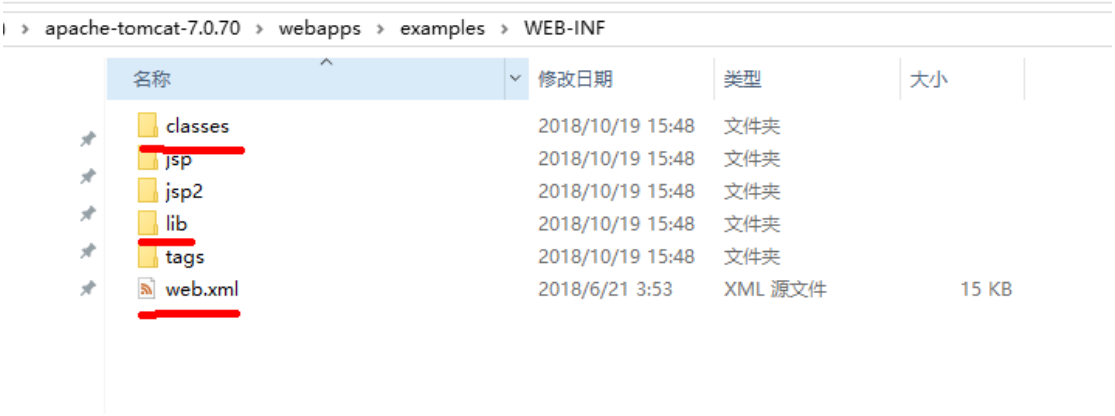
D) html 文件

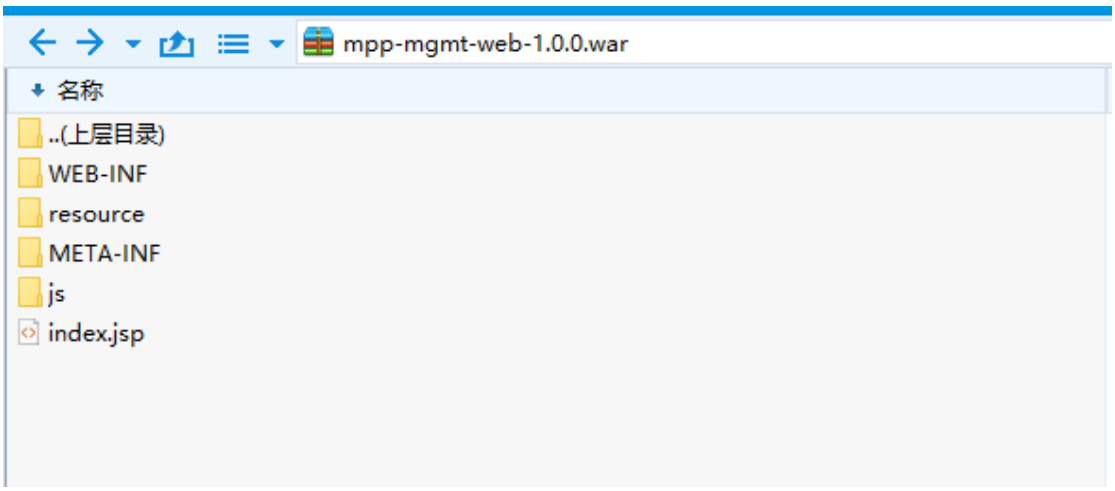
解析:

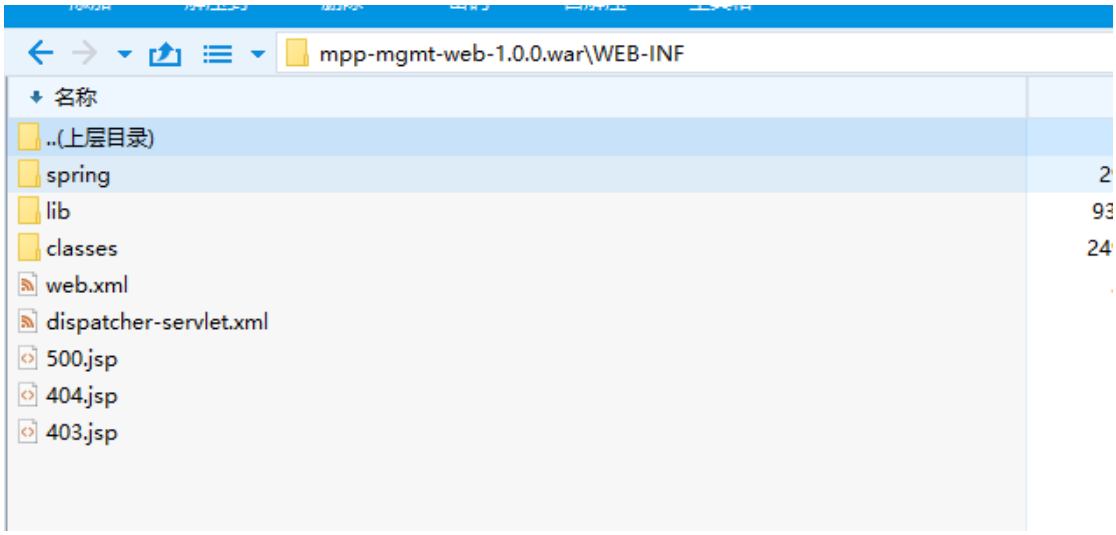
开发过程中的目录结构:


mpp-mgmt-web 1.0.0 [app-mpp/branches/20181117/webapps/mpp-mgmt-web]
src/main/java
src/main/resources
src/test/java
JRE System Library [JavaSE-1.6]
Maven Dependencies
src
main
webapp
js
resource
WEB-INF
spring
403.jsp
404.jsp
500.jsp
dispatcher-servlet.xml
web.xml
index.jsp
test
target
pom.xml

部署过程中的目录结构:


apache-tomcat-7.0.70 › webapps › examples › WEB-INF
名称
修改日期
类型
大小
classes 2018/10/19 15:48 文件夹
jsp 2018/10/19 15:48 文件夹
jsp2 2018/10/19 15:48 文件夹
lib 2018/10/19 15:48 文件夹
tags 2018/10/19 15:48 文件夹
web.xml 2018/6/21 3:53 XML 源文件 15 KB


mpp-mgmt-web-1.0.0.war
名称
..(上层目录)
WEB-INF
resource
META-INF
js
index.jsp



4. 查找效率最高的二叉排序树是（）

A) 所有节点的左子树都为空的二叉排序树

B) 所有节点的右子树都为空的二叉排序树

C) 平衡二叉树

D) 没有左子树的二叉排序树

解析:

二叉查找数的查找速度取决于树的深度,相同节点数深度最小的是平衡二叉树.

https://www.nowcoder.com/questionTerminal/7e46cac41b0f42d3927e41921b5cb606?source=relative

来源：牛客网

对于一般的二叉搜索树（Binary Search Tree），其期望高度（即为一棵平衡树时）为 log2n，其各操作的时间复杂度（O(log2n)）同时也由此而决定。但是，在某些极端的情况下（如在插入的序列是有序的时），二叉搜索树将退化成近似链或链，此时，其操作的时间复杂度将退化成线性的，即 O(n)。我们可以通过随机化建立二叉搜索树来尽量的避免这种情况，但是在进行了多次的操作之后，由于在删除时，我们总是选择将待删除节点的后继代替它本身，这样就会造成总是右边的节点数目减少，以至于树向左偏沉。这同时也会造成树的平衡性受到破坏，提高它的操作的时间复杂度。

平衡二叉搜索树（Balanced Binary Tree）具有以下性质：它是一棵空树或它的左右两个子树的高度差的绝对值不超过 1，并且左右两个子树都是一棵平衡二叉树。常用算法有红黑树、AVL、Treap、伸展树等。在平衡二叉搜索树中，我们可以看到，其高度一般都良好地维持在 O（log2n），大大降低了操作的时间复杂度。

5. Linux 查看文件列表，应使用（）命令。

A) cat

B) more

C) less

D) ls

解析:

拓展阅读: http://www.runoob.com/linux/linux-command-manual.html

---

cat:用于连接文件并打印到标准输出设备上

注意：当文件较大时，文本在屏幕上迅速闪过（滚屏），用户往往看不清所显示的内容。因此，一般用 more 等命令分屏显示。为了控制滚屏，可以按 Ctrl+S 键，停止滚屏；按 Ctrl+Q 键可以恢复滚屏。按 Ctrl+C（中断）键可以终止该命令的执行，并且返回 Shell 提示符状态。

---

more: 是一个基于 vi 编辑器文本过滤器，它以全屏幕的方式按页显示文本文件的内容，支持 vi 中的关键字定位操作.

more 名单中内置了若干快捷键，常用的有 H（获得帮助信息），Enter（向下翻滚一行），空格（向下滚动一屏），Q（退出命令）。

该命令一次显示一屏文本，满屏后停下来，并且在屏幕的底部出现一个提示信息，给出至今已显示的该文件的百分比：--More--（XX%）可以用下列不同的方法对提示做出回答：

按 Space 键：显示文本的下一屏内容。

按 Enier 键：只显示文本的下一行内容。

按斜线符|：接着输入一个模式，可以在文本中寻找下一个相匹配的模式。

按 H 键：显示帮助屏，该屏上有相关的帮助信息。

按 B 键：显示上一屏内容。

按 Q 键：退出 rnore 命令

---

less 命令的作用与 more 十分相似，都可以用来浏览文字档案的内容，不同的是 less 命令允许用户向前或向后浏览文件，而 more 命令只能向前浏览

用 less 命令显示文件时，用 PageUp 键向上翻页，用 PageDown 键向下翻页。要退出 less 程序，应按 Q 键

---

ls 命令用来显示目标列表，在 Linux 中是使用率较高的命令。ls 命令的输出信息可以进行彩色加亮显示，以分区不同类型的文件

6. 假设有一段代码 Thread.sleep(3000),那么在多久之后该代码能获取到对 CPU 的控制权(假设该时间内没有其他优先级更高的线程到准备就绪状态)

A) >3000ms

B) <3000ms

C) =3000ms

D) 不一定

7. 下面关于垃圾收集的说法正确的是

A) 一旦一个对象成为垃圾，就将立即被回收掉

B) 对象空间被回收掉之后，就将执行该对象的 finalize()方法

C) finalize()方法和 C++的析构函数是完全一回事

D) 一个对象成为垃圾是因为不再有引用指向它，而线程并非如此

解析:

A: 等 GC 执行完成

B:java 对象在标记为垃圾回收时调用此方法,而不是空间被回收之后

C:来自 Thinking in Java

java 提供 finalize()方法，垃圾回收器准备释放内存的时候，会先调用 finalize()。

(1).对象不一定会被回收。

(2).垃圾回收不是析构函数。

(3).垃圾回收只与内存有关。

(4).垃圾回收和 finalize()都是靠不住的

8. 下面哪个是属于面向字符的输入流？

A) BufferedWriter

B) FileInputStream

C) ObjectInputStream

D) InputStreamReader

解析:

流的分类

按数据流的方向不同：输入流，输出流。

按处理数据单位不同：字节流(**以 Input/OutputStream 结尾**)，字符流(**以 Reader/Writer 结尾**)。

（1） 字节流：数据流中最小的数据单元是字节。

（2）字符流：数据流中最小的数据单元是字符， Java 中的字符是 Unicode 编码，一个字符占用两个字节。

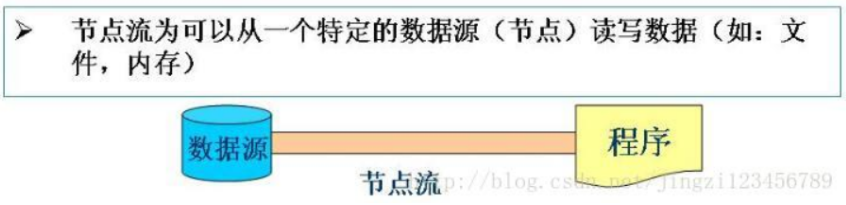
按功能不同：节点流，处理流。

（1）程序用于直接操作目标设备所对应的类叫节点流。

（2）程序通过一个间接流类去调用节点流类，以达到更加灵活方便地读写各种类型的数据，这个间接流类就是处理流。

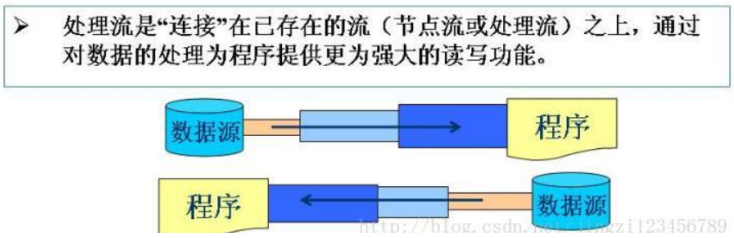
节点流

类 型	字 符 流	字 节 流
File（文件）	FileReader FileWriter	FileInputStream FileOutputStream
Memory Array	CharArrayReader CharArrayWriter	ByteArrayInputStream ByteArrayOutputStream
Memory String	StringReader StringWriter	—
Pipe（管道）	PipedReader PipedWriter	PipedInputStream PipedOutputStream



处理流

处理类型	字符流	字节流
Buffering	BufferedReader BufferedWriter	BufferedInputStream BufferedOutputStream
Filtering	FilterReader FilterWriter	FilterInputStream FilterOutputStream
Converting between bytes and character	InputStreamReader OutputStreamWriter	
Object Serialization	—	ObjectInputStream ObjectOutputStream
Data conversion	—	DataInputStream DataOutputStream
Counting	LineNumberReader	LineNumberInputStream
Peeking ahead	PusbackReader	PushbackInputStream
Printing	PrintWriter	PrintStream



9. 关于 touda 服务开发，说法错误的有（）

A) **Touda 日志打印允许第三方日志 api**

B) 渠道服务命名规范为渠道类型（J/X）+_+交易码，不得超过 20 字符

C) 渠道类型简称 J 代表 json 格式报文，X 代表 SOAP 格式报文

D) Touda 服务在引用第三方 jar 包时优先考虑 toudaServer 已经依赖的 jar 包

解析:

参考 touda 开发规范

2.9.2 日志打印

1、在 Touda 中的日志打印必须使用 Touda 提供的日志输出 API。

```
import com.primeton.btp.api.core.logger.ILogger;
import com.primeton.btp.api.core.logger.LoggerFactory;

private static final ILogger logger = LoggerFactory.getLogger(Abc.class);
```

10. 下列代码多线程调用不需要加同步块的（）

A) x=y

B) x++

C) ++x

D) x=1

11. 关于 touda 交易流，下列描述错误的是（）

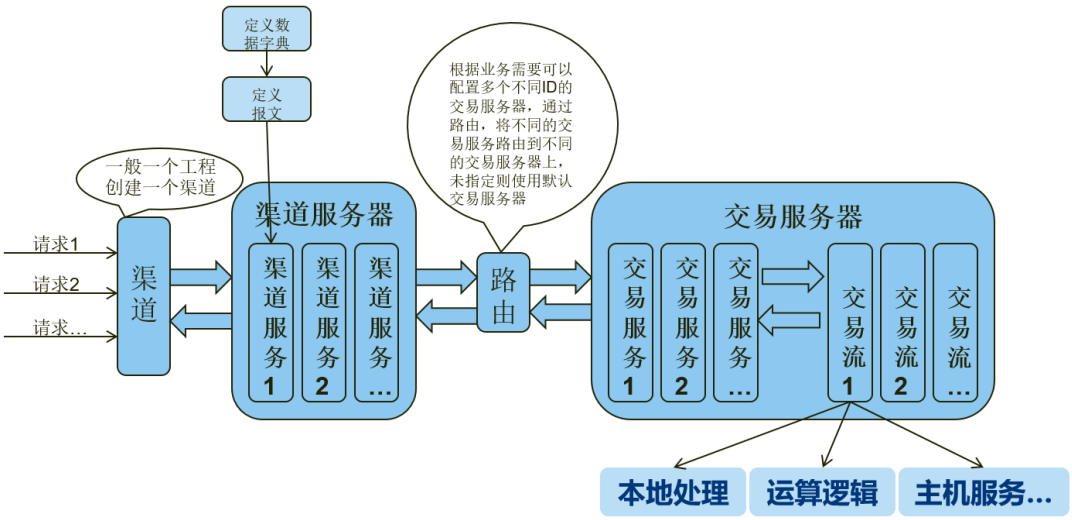
A) 交易流中可以直接调用本地处理

B) 交易流中可以直接调用运算逻辑

C) 交易流中可以直接调用渠道服务

D) 交易流中可以直接调用主机服务

解析:



12. toudaIDE 创建新空间是需要进行一些必要的设置，下列设置中错误的是（）

A) PrimetonEos 设置，包含作者，部门，默认 Module 名称设置

B) 设置工作空间编码格式为 UTF-8

C) Touda 项目使用人 JRE 版本需要设置为 1.6 以上

D) 编译器一致性级别需要设置为 6.0

解析:

JRE 要求 1.7 以上.即使是 touda3.0 也即是如此

以上这 3 点是创建 touda 工程的必须做的配置项

13. 以下关于外键的说法正确的是（）

A) 外键必须和相应的主键同名

B) 外键可以和相应的主键不同名，只要定义在相同的域上即可

C) 外键值不可以为空

D) 外键的取值只允许等于所参照关系中的某个主键值

解析:

主要考查的知识点为参照完整性规则。

外键和相应的主键只要定义在相同的值域上即可，不必同名选项 A 错

外键值是否为空要视具体的情况区别对待，故选项 C/D 错误。

14. 关于 toudaDFS,下列说法错误的是（）

A) toudaDFS 定位容量小文件存储和高并发访问

B) toudaDFS 不支持在非 touda 应用中使用

C) toudaDFS 实现了软件方式的 RAID，提高了磁盘的利用率

D) DFS 支持存储服务器在线扩容

解析:

ToudaDFS 定位**海量小文件存储**和**高并发访问**，实现了软件方式的 RAID，提高了磁盘的利用率.

另 1 考点:临时文件默认保留 30 天.除非在调用文件上传 api 时显式的声明文件为永久保留

主要的调用方就是非 touda 应用.所以 B 选项错误.

15. 关于 touda 分布式服务框架，下列描述错误的是（）

A) touda 是支撑应用从架构设计，业务建模，服务编排，应用发布到运行管理的一体化平台化的解决方案和分布式服务架构

B) touda 采用 SEDA 架构实现了“多服务接入，多核心服务，多主机接出”技术架构

C) touda 符合微服务架构的原则，支持“服务开发，服务发布，运行管理”等一系列开发配套

D) touda 应用集成前端展现，服务端处理，数据库一体，开发，测试，部署非常方便

解析:

touda 不支持前端展现功能

16. 关于 touda 项目的说法，错误的是（）

A) 创建项目时，可根据实际情况，选择创建架构设计构建包

B) 项目中必须包含业务建模构建包

C) 项目中必须包含服务定制模块

D) 本地进行项目开发时，需要引入 function 工程

解析:**(本题关于 function 工程是否需要导入,存在争议)**

3.0 版本 IDE 在创建新项目时,默认创建 4 个包

1.0 版本 IDE 需要主动创建 EOS 构件包(3.0 版本改名叫服务逻辑).

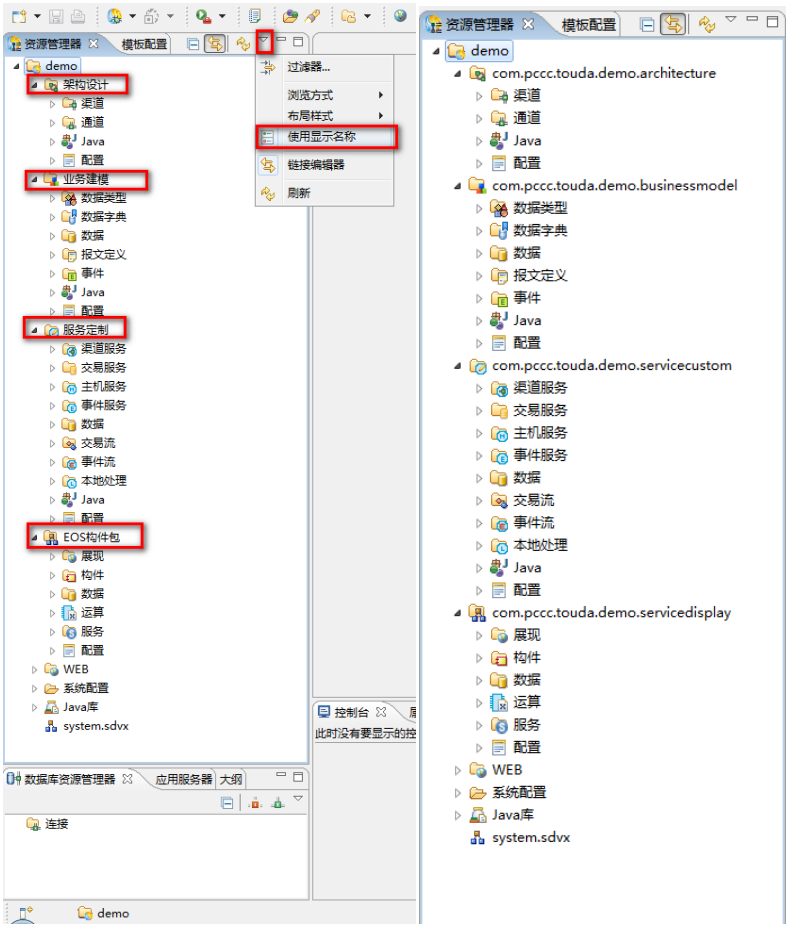
无论什么版本,架构设计/业务建模/服务定制这 3 个都是默认创建的

争议部分:

touda1.0 版本:开发主机服务时需要导入 function 工程

touda3.0 版本:已经无需主动导入,在 jenkins 编译时会自动依赖进去.

解决办法:1:找基础平台组确认.2:如果没有合适的选项的话,选它,认为他是需要依赖的.依赖没有什么毛病.只是不需要强制依赖了.



17. 关于云上应用，下列说法错误的是（）

A) 云上应用日志不允许输出到本地，查看日志需要登录统一日志平台查看

B) 新增的微应用，需要走双速流程申请，取得合法应用名

C) tomcat 应用上云配置项无需在配置中心进行管理

D) touda 应用上云，渠道暴露端口必须为 8080

解析:

按照上云的规范文档说明,所有的配置项必须在配置中心配置.

理解:

在 jenkins 编译过程中,编译脚本会把项目自带的 touda.properties 文件移除.

在 jenkins 部署过程中,部署脚本会从配置中心拉取配置,将生成 touda.properties

18. 关于 logback 日志打印级别，正确的是（）

A) TRACE<DEBUG<INFO<WARN<ERROR

B) DEBUG<TRACE<INFO<WARN<ERROR

C) TRACE<DEBUG<WARN<INFO<ERROR

D) TRACE<DEBUG<INFO<ERROR<WARN

19. 下面配置项命名规范的是

A) hulianwangtransaction

B) next_host

C) touda.proxy.host

D) Jiaotongbank

解析:

遵循阿里巴巴 Java 开发规约.

1. touda.properties
   配置项管理
   a) 配置项 key 命名规范
      key 中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。
      说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式也要避免采用。
      正例：alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。
      反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3

   b) 配置项 key 命名分割
      key 中的命名涉及某个功能或者某个组件的的多个配置，采用"."进行分割，对于无法分割的词语，使用驼峰风格，遵从驼峰形式
      正例：marcoPolo /touda.proxy.server.port

   c) 配置项 value 换行问题
      如果配置项的 value 需要换行，但请使用\进行连接
      key=value\value

   d) touda.properties 中 touda.app.id 配置项
      必填且不可变更，初始值需要向基础平台组申请，即服务发布申请时该项目对应的应用 ID

   e) touda.properties 中 touda.zk.address 配置项

环境	配置
SIT	touda.zk.address=182.180.80.102:2181
UAT	touda.zk.address=182.180.115.33:2181,182.180.115.34:2181,182.180.115.35:2181

20. 关于 touda 渠道服务，描述错误的是（）

A) 渠道服务处理报文打解包功能

B) 渠道服务可以对报文的输入进行校验

C) 渠道服务可以配置交易事务

D) 渠道服务可以控制交易路由

解析:

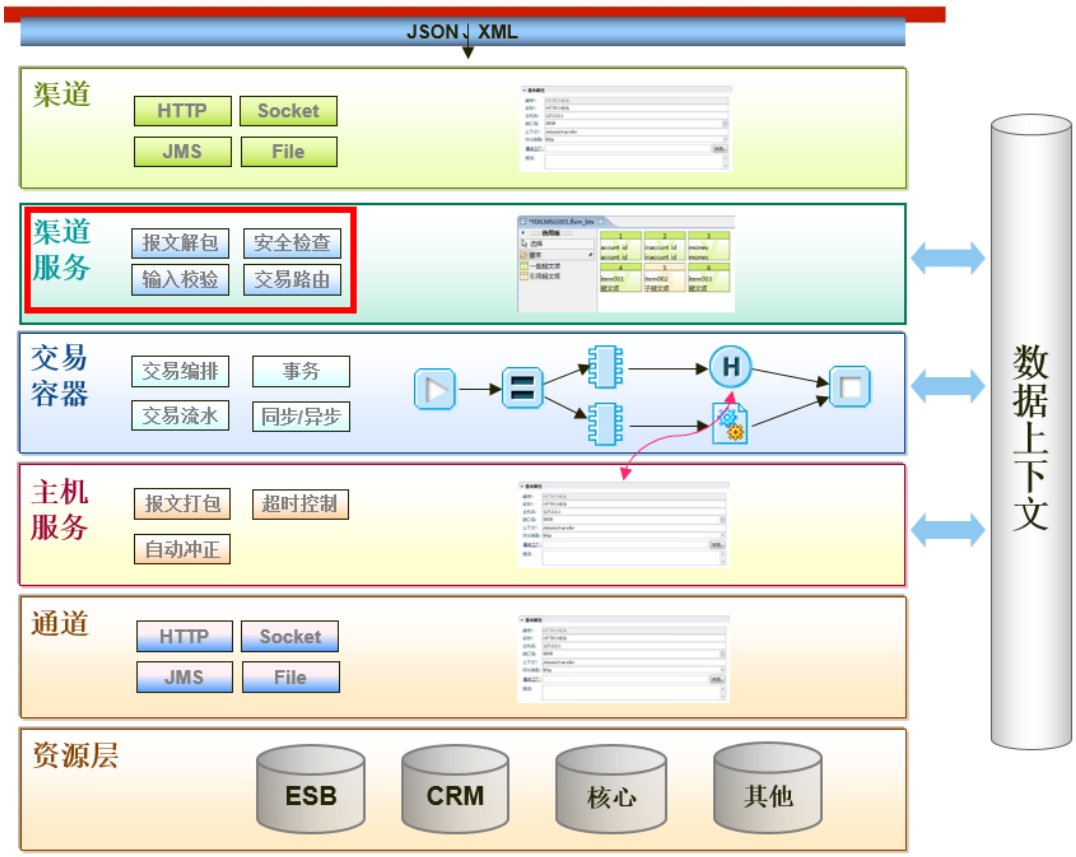
渠道：处理协议层的东西，接受外部调用，支持 http、socket、JMS、File 等

渠道服务：报文打/解包（请求进来时报文解包，响应时报文打包）、安全检查、输入校验、交易路由

交易容器（交易服务器）：交易编排、事务、交易流水、同步/异步

主机服务（与渠道服务对应）：报文打包、超时控制（保证提供确定的服务）、自动冲正

通道（与渠道对应）：处理协议层的东西，调用外部服务



21. 关于熔断器，下列描述错误的是（）

A) 在高并发访问下，熔断器能够隔离一些不可控因素，从而保证系统的稳定

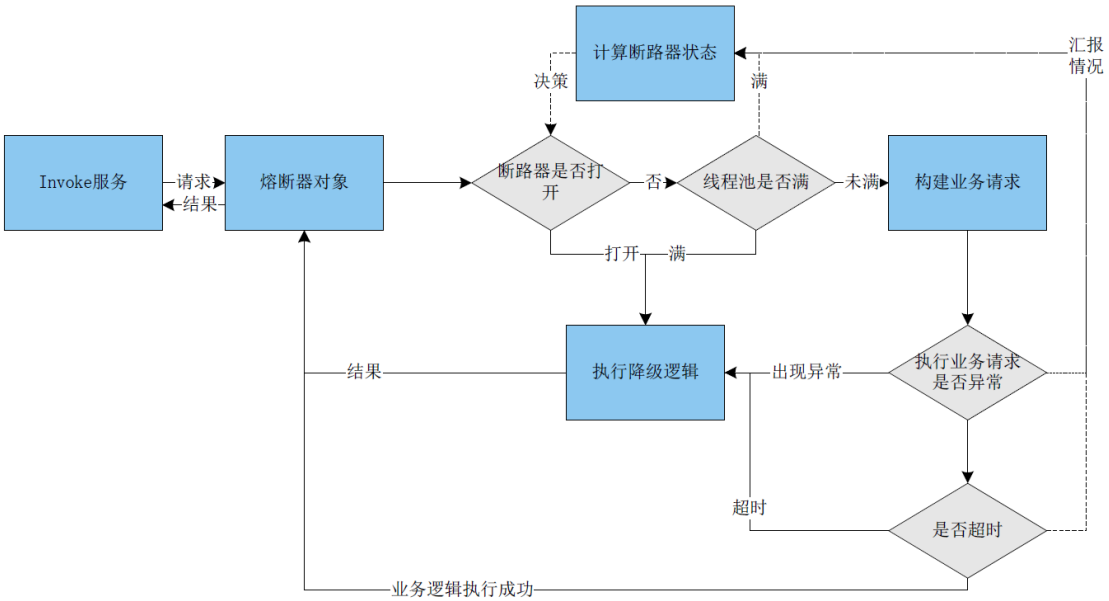
B) 熔断器使用断路器和线程池来决定所有的请求是否执行降级策略

C) 使用熔断器功能需要在服务治理中心下熔断器管理页面配置具体的熔断器

D) 熔断器功能可以在 touda1.0 版本下运行

解析:

熔断器是在 3.0 版本新增的功能.



22. 关于服务注册中心，下面说法错误的是（）

A) Touda 服务提供方应用启动时向 touda 注册中心注册服务信息

B) Touda 服务调用方在每次调用时，从 touda 注册中心拉去服务订阅列表信息

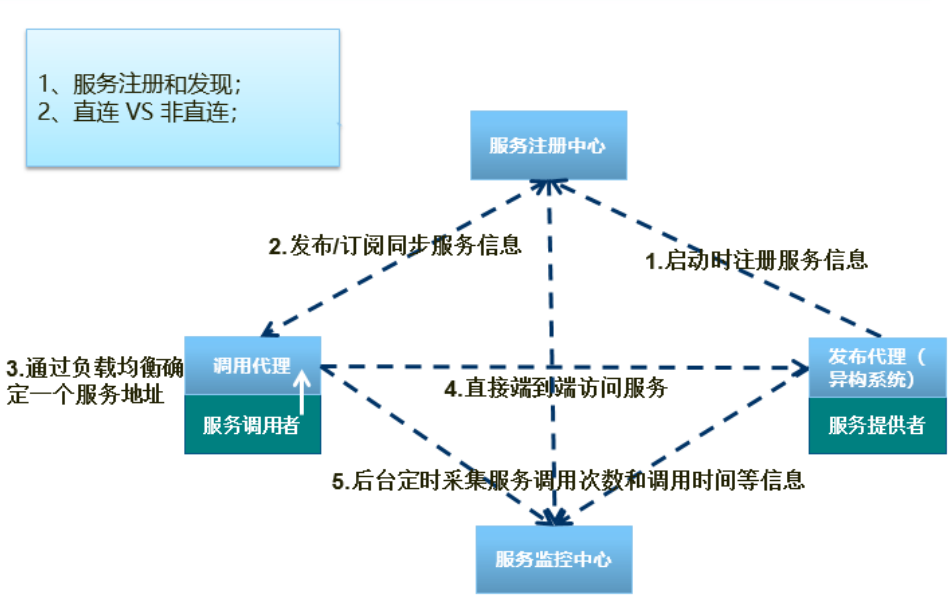
C) Touda 服务注册中心会定时向 touda 监控中心上报监控信息

D) Touda 服务实际调用时为端到端访问，请求不会经过第三方

解析:

肯定不会每次调用都去拉取,否则系统性能都完全消耗在辅助功能上了.

熟记服务注册中心的图



23. 关于 touda 服务治理中心，下列说法错误的有（）

A) touda 服务治理中心主要管理 touda 相关的服务

B) touda 服务治理中心可以对服务进行降级操作

C) touda 服务治理中心可以对服务提供者进行禁用操作

D) touda 服务治理中心不能管理非 touda 开发的服务

24. 关于 touda 平台的日志，下列说法错误的是（）

A) touda-system.log 是系统的边界日志，即记录了接收到的请求报文和返回的响应报文

B) touda-trace.log 是记录系统调试日志，用于系统维护人员定位系统运行问题使用

C) touda-proxy.log 记录执行 sql 脚本情况，包括执行 sql 时入参值和响应时间等信息

D) 日志文件中的全局流水号唯一标识一次完整的交易

解析:

Touda 核心日志文件包括以下几种

touda-proxy.log 主要记录“com.pccc.touda.proxy”包下的日志信息，默认是 Info 级别，主要用于查看通过 Invoke 调用服务的日志信息。

touda-trace.log 记录各自应用打印的日志，也包括“com.pccc.touda”包下的所有日志，默认是 info 级别。

touda-system.log 主要记录渠道、通道的请求报文及响应报文信息。属于边界日志.

touda-engine.log 日志主要是记录执行的 sql 脚本情况，数据库连接，执行 sql

touda-app.log 主要记录开发人员用 logger 打印的日志.生产环境固定打印 ERROR 级别的日志

touda-deploy.log 主要记录 touda 应用启动的日志信息

25. touda 服务发布流程的正确顺序是

1. 生产发布信息初审：应用运维负责人负责审核申请表,补充生产环境信息
2. 等待 touda 负责人确认：touda 负责人审核开发提交的 excel 模板，若无误，则录入到 uat 环境的服务治理中心，否则，驳回
3. 等待 uat 测试：服务信息已经录入到 uat 环境，开发进行非直连方式代码验证
4. 生产发布信息复审：touda 运维人员审核各个组运维人员补充生产信息后提交的申请单
5. 等待投产：touda 运维在制定时间根据 excel 模板将服务录入到生产服务治理中心
6. 已投产：服务已经录入到生产服务治理中心
7. 初始：开发完成服务发布申请相关模板，提交申请

A) 7321456

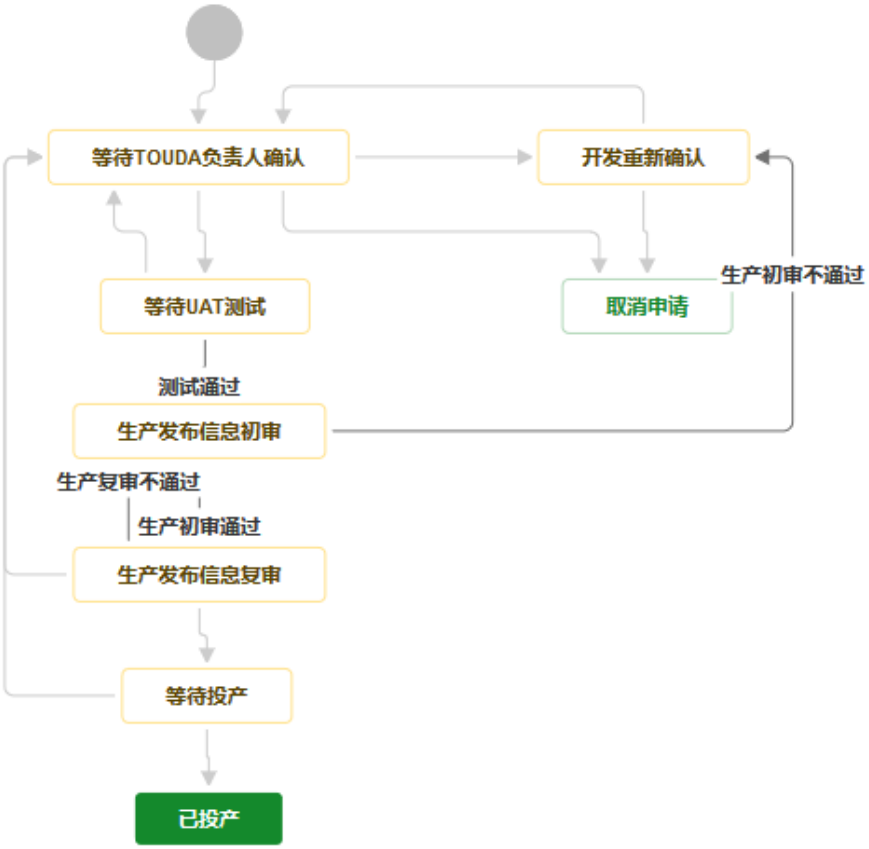
B) 7231456

C) 7235146

D) 7325146

解析:

按照我们实际使用的流程来记忆,不用强行记忆都行.



多选

1. 以下哪几个能使用 throw 抛出

   A) Error

   B) Event

   C) Object

   D) Throwable

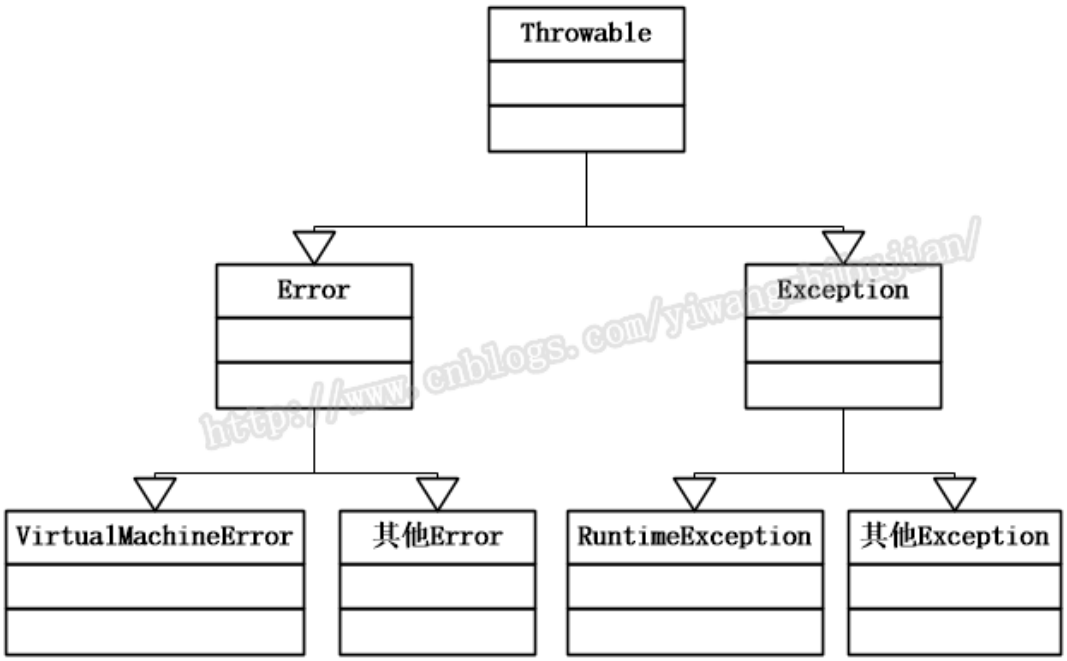
   E) Exception

   F) RuntimeException

   解析:

   其实是考查 Java 中的异常体系

   凡是 Throwable 的子类都可以被 throw 关键字抛出



2. 不能用来修饰 interface 的有（）

   A) private

   B) public

   C) protected

   D) static

   解析:(本题其实有争议,但是在这个场景下可以选择出正确答案)

   如果 interface 是做为一个单独的类文件来说,只能用 public 修饰,或者不要修饰符

InterfaceTest.java | InterfaceTest1.java

```java
package com.pccc.piap.mall.common.config;

public interface InterfaceTest1 {

}
```

InterfaceTest.java | InterfaceTest1.java

```java
package com.pccc.piap.mall.common.config;

public class InterfaceTest {

    static interface InnerInterface1 {

    }

    private interface InnerInterface2 {

    }

    protected interface InnerInterface3 {

    }

    public interface InnerInterface4 {

    }

}
```

3. 开发一个完整的 touda 服务，下列说法正确的有（）

A) 每个 touda 服务必须单独创建一个渠道

B) 每个 touda 服务必须单独创建一个渠道服务

C) 每个 touda 服务必须单独创建一个主机服务

D) touda 服务的请求报文和响应报文的定义依赖数据字典

解析:

渠道只需要创建一次.每个应用只可以有 1 个 HTTP 渠道.

主机服务按照需要创建.

渠道服务=接口.

渠道服务:交易服务:交易流:请求报文:响应报文=1:1:1:1:1

4. 下列关于 touda 事务控制，说法正确的有（）

A) touda 事务可以由框架控制，也可以再交易流中自己控制事务

B) touda 本地事务是指数据库级别的事务

C) touda 支持全局事务

D) 用了 touda 本地事务，在交易流中仍然可以自己局部控制事务

5. 关于 touda 的配置文件，说法正确的有

A) user-config.xml 是 touda 应用的数据源配置文件

B) logback.xml 是日志相关配置文件，可配置日志输出到哪个文件，以及日志打印格式等等

C) touda.properties 仅对 touda 应用是必要的，对 web 应用和批处理应用而言可有无

D) spring-proxy.xml 是调用代理配置文件

解析:

凡 touda 服务端或者客户端(即:你的应用需要调用 touda 接口),都需要依赖 touda.properties

下图 1 声明了 1 个名为 default 的数据源,在 user-config.xml 中配置

且需要在服务定制的配置目录中的 contribution.eosinf 文件中声明(下图 2)

```
-->
<module name="DataSource">
    <!--system default datasource -->
    <group name="default">
        <configValue key="C3p0-DriverClass">com.ibm.db2.jcc.DB2Driver</configValue>
        <configValue key="C3p0-Url">jdbc:db2://182.180.80.16:50000/touda_db</configValue>
        <configValue key="C3p0-UserName">toudausr</configValue>
        <configValue key="C3p0-Password">1qaz@WSX</configValue>
        <configValue key="C3p0-PoolSize">10</configValue>
        <configValue key="C3p0-MaxPoolSize">50</configValue>
        <configValue key="C3p0-MinPoolSize">10</configValue>
        <configValue key="Transaction-Isolation">ISOLATION_READ_COMMITTED</configValue>
        <configValue key="Database-Type">DB2</configValue>
        <configValue key="Jdbc-Type">IBM DB2 Driver(Type4)</configValue>
        <configValue key="Test-Connect-Sql">SELECT count(*) from EOS_UNIQUE_TABLE</configValue>
        <configValue key="Retry-Connect-Count">-1</configValue>
    </group>
    <!--
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<contribution xmlns="http://www.primeton.com/xmlns/eos/1.0">
    <!-- MBean config -->
    <module name="Mbean">
        <!-- DataSourceMBean config -->
        <group name="DatasourceMBean">
            <configValue key="Type">config</configValue>
            <configValue key="Class">com.eos.system.management.config.mbean.Con
            <configValue key="Handler">com.eos.common.connection.mbean.Contribu
            <configValue key="ConfigFileType">config</configValue>
        </group>
        <group name="ContributionLoggerMBean">
            <configValue key="Type">config</configValue>
            <configValue key="Class">com.eos.system.management.config.mbean.Con
            <configValue key="Handler">com.eos.common.logging.mbean.LogConfigHa
            <configValue key="ConfigFileType">log</configValue>
        </group>
    </module>

    <!-- datasource config -->
    <module name="DataSource">
        <group name="Reference">
            <!--
                the configuration below describes
                the corresponding relationship between contribution datasource
                multiple datasources can be defined.
                the value 'default' of attibute 'key' denotes a contribution da
                and the field value 'default' of 'configValue' node stands for
            -->
            <configValue key="default">default</configValue>
        </group>
    </module>
</contribution>
```

6. java 多线程有几种实现方法（）

   A) 继承 Thread 类

   B) 实现 Runable 接口

   C) 实现 Thread 接口

   D) 实现 Callable 接口

   E) 以上都不正确

   解析:

   Java 多线程实现的方式有四种

   1.继承 Thread 类，重写 run 方法

   2.实现 Runnable 接口，重写 run 方法，实现 Runnable 接口的实现类的实例对象作为 Thread 构造函数的 target

   3.通过 Callable 和 FutureTask 创建线程

   4.通过线程池创建线程

   ==========

   前面两种可以归结为一类：无返回值，原因很简单，通过重写 run 方法，run 方式的返回值是 void，所以没有办法返回结果

   后面两种可以归结成一类：有返回值，通过 Callable 接口，就要实现 call 方法，这个方法的返回值是 Object，所以返回的结果可以放在 Object 对象中

   拓展阅读: https://blog.csdn.net/u011480603/article/details/75332435/

7. 下面哪几个描述是正确的（）

   A) 默认构造器初始化方法变更

B) 默认构造器有和它所在类相同的访问修饰词

C) 默认构造器调用其父类的无参构造器

D) 如果一个类没有无参构造器，编译器会为它创建一个默认构造器

E) 只有当一个类没有任何构造器时，编译器会为它创建一个默认构造器

8. 下面的对象创建方法中哪些会调用构造方法（）

A) new 语句创建对象

B) 调用 Java.io.ObjectInputStream 的 readObject 方法

C) java 反射机制使用 java.lang.Class 或 java.lang.reflect.Constructor 的 newInstance()方法

D) 调用对象的 clone()方法

9. 下面哪项技术可以用在 WEB 开发中实现会话跟踪实现（）

A) session

B) Cookie

C) 地址重写

D) 隐藏域

解析:

https://www.nowcoder.com/questionTerminal/8d86bb29952642169e816d519f8417f3?orderByHotValue=1&pos=15

Cookies

Cookies 是使用最广泛的会话跟踪机制，Cookies 是有服务器创建，并把 Cookies 信息保存在用户机器上的硬盘上，下次用户再次访问该站点服 务器的时候，保存在用户机器上硬盘的 Cookies 信息就被送回给服务器。一般 Cookies 一般不多于 4KB，且用户的敏感信息如信用卡账号密码不应该 保存在 Cookies 中。

URL 重写

URL 重用户在每个 URL 结尾附加标识回话的数据，与标识符关联的服务器保存有关与会话的数据，如我们访问某个新闻的时候，在地址栏我们一般会看到这样的 信息：http://www.XXX.com/news?id=??,通常的话 id 后面的问号表示该条新闻在后台数据库中的新闻表的 id。URL 重写能够 在客户端停用 cookies 或者不支持 cookies 的时候仍然能够发挥作用。

隐藏表单域

通常，在表单中我们使用隐藏表单域的时候会有这么一句代码：<input type=”hidden” name=”XXX” value=” XXX” />。通过给 type 属性赋值为 hidden 值来实现隐藏，这样用户在浏览的时候看不到这行代码的数据，但是当用户通过查看 源代码还是可以看到的。

Session 机制

这个机制要慎用，特别是对于访问量很大的站点，因为这种机制是吧 Session 信息保存在服务器端。如果访问量特别大的话，对于服务器的承受力的要求有多高是可想而知的。

10. 下列说法正确的是（）

A) toudaServer 部署依赖 Tomcat 等运行容器

B) 一台服务器可以部署多个 toudaServer；

C) touda 提供的负载均衡功能是在服务调用方法完成的

D) touda 服务提供者信息发生变更时，变更内容由 touda 注册中心通知服务调用方

解析:

部署架构是无容器：不依赖其他容器，只是个 JVM 进程，一台服务器上可以部署多个 ToudaServer

touda 服务提供者信息发生变更时,由订阅方收到通知,并主动从 ZK 中获取最新的数据.

简答

1 代理模式是 JAVA 中一种常见的设计模式，请写出一段动态代理模式代码

解析:

参考 http://www.runoob.com/design-pattern/proxy-pattern.html

参考附件

Java动态代理实现与原理详细分析.do

2 写两个线程，第一个线程打印 1-52，第二个线程打印 A-Z，打印结果为 12A34B...5152Z

解析:

参考附件

https://blog.csdn.net/wallance111/article/details/50864071

1、创建两个线程实现 Runnable 接口重写 run 方法，一个用于打印数字，一个用于打印字母。

2、创建一个测试类，在测试类中创建一个 Object 类的对象（作为两个线程的共享资源，以便实现线程间的通信），通过各类的构造方法传递过去。

3、在两个类的 run 方法中都要用 synchronized 保证同步，即加锁。

4、在数字类中用 for 循环每打印两个数字就唤醒其他线程，释放锁，进入阻塞状态。

在字母类中每打印一个字母就唤醒其他线程，释放锁，进入阻塞状态。

在写这个程序的时候有几点要注意的地方：

1、两个线程要使用同一个资源才需相互通信，所以在测试类中创建共享资源，并通过构造方法分别传到各线程类中。

2、两个线程哪个先运行（执行 start（））哪个就先获得资源并执行

3、在 run 方法体内写进程间的通信 wait（）和 notifyall（）时，一定要先写 notifyall（）再写 wait（）。

原因：当你先写 wait()时，本进程也进入休眠状态，再写 notifyall（）唤醒所有线程时本线程以及其他线程被一块唤醒，竞争同一个资源，就会造成死锁。

所以一定要先唤醒其他线程，再让本线程阻塞

```
public class ThreadPrinter {
    // true打印数字，false打印字母
    private boolean flag = true;

    public synchronized void printNumber(String s) {
        try {
            if (!flag) {
                super.wait();
            }
```

```
            System.out.print(s);
            flag = false;
            super.notifyAll();
        } catch (InterruptedException ex) {
        }
    }

    public synchronized void printLetter(String s) {
        try {
            if (flag) {
                super.wait();
            }
            System.out.print(s);
            flag = true;
            super.notifyAll();
        } catch (InterruptedException ex) {
        }
    }

    public static class LetterPrintThread extends Thread {
        public ThreadPrinter printer;

        public LetterPrintThread(String name, ThreadPrinter printer) {
            super(name);
            this.printer = printer;
        }

        public void run() {
            for (int i = 1; i <= 26; i++) {
                int temp = i + 64;
                char c = (char) temp;
                printer.printLetter(c + " ");
            }
        }

    }

    public static class NumberPrintThread extends Thread {
        public ThreadPrinter printer;

        public NumberPrintThread(String name, ThreadPrinter printer) {
            super(name);
            this.printer = printer;
        }
```

```
        public void run() {
            for (int i = 1; i <= 26; i++) {
                String s = (2 * i - 1) + " " + 2 * i + " ";
                printer.printNumber(s);
            }

        }
    }

    public static void main(String[] args) {
        ThreadPrinter printer = new ThreadPrinter();
        new NumberPrintThread("打印数字线程", printer).start();
        new LetterPrintThread("打印字母线程", printer).start();
    }
}
```